User's guide to the VAMDC registry

Document Information

Editors:	L. Nenadovic, M. Doronin
Authors:	K. Benson, G. Rixon
Contributors:	M. Doronin
Type of document:	software documentation
Status:	draft
Distribution:	public
Work package:	N/A
Version:	12.07
Date:	06/08/2012
Document code:	
Document URL:	http://www.vamdc.org/documents/registry-guide_v12.07.pdf

Abstract: This document is a guide to the VAMDC registry. It describes the administration pages, how to query the registry and how to register new VAMDC-TAP services.

Version History

Version	Date	Modified By	Description of Change
V0.1	03/03/2011	L. Nenadovic	inserted document into template
V0.2	09/03/2011	K. Benson	combined pdf user guide with document
V0.3	09/03/2011	L. Nenadovic	formatted headings, table of contents and links
V0.4	22/04/2011	L. Nenadovic	changed capitalisation for consistency
V11.5	29/05/2011	L. Nenadovic	insert into Sphinx framework
V11.12	21/12/2011	G. Rixon	update the guide to v11.12
V12.07	06/08/2012	M.Doronin	update registry addresses for 12.07 system

Disclaimer

The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

License

This document reflects only the authors' view. The European Community is not liable for any use that may be made of the information contained herein.

Acknowledgements

CONTENTS

INTRODUCTION

The International Virtual Observatory Alliance (IVOA) registry allows astronomers to search, obtain details of, and leverage any of the resources located anywhere in the IVO space, namely in any Virtual Observatory. The IVOA defines the protocols and standards whereby different registry services are able to interoperate and thereby realise this goal. IVOA registry defines interfaces on how to query and share resources. Software is written to conform to standard interfaces in order to assist scientific utilities to access particular resource. A resource in this context is represented in XML form and is stored in the registry. A resource may describe anything about an observatory, particular instrument, another registry, and services such as catalogue or table type services, cone searches. Extensions can be made if necessary and this functionality is made available for VAMDC.

1.1 Resource definition

Resources conform to a standard schema and every XML request is validated to the schema before it can be submitted to the registry for querying. More information on IVOA schemas can be found here: http://www.ivoa.net/xml/index.html

1.2 VOResource and Dublin Core

XML resources derive from a common top layer schema titled 'VOResource'. The VOResource may also be referred to as 'Core' or 'Dublin Core' as it contains the complete set of the necessary core data. More information on VOResource documentation can be found here: http://www.ivoa.net/Documents/REC/ReR/VOResource-20080222.html

1.3 Identifier

Every resource in the registry must have an identifier (similar a primary key), which is URI based. A sample: `ivo://vamdc/chianti/chianti_catalogue_service`

Identifier must be in the following form:

`ivo://{authorityid}/{resourcekey}`

Registry manages authorityIDs. Any other registry cannot duplicate an authorityID, it is owned by one registry only. For the purposes of VAMDC only one authority id `vamdc` is managed at present. ResourceKey is a localised name and is unique in respect of the authorityID.

Though an identifier can be of any form, it is widely accepted that authorityID is a domain name or a subsection of an institute, such as `mssl.ucl.ac.uk` or `climatephysics.mssl.ucl.ac.uk`. Currently the assumption has been made that VAMDC only needs one main registry and will use the authority ID `vamdc`. A ResourceKey is typically a name with reference to the registered resource.

More information about registry identifiers can be obtained below http://www.ivoa.net/Documents/REC/Identifiers/Identifiers-20070302.html

1.4 Virtual Observatory Support Interface (VOSI)

The Support interface is required by all IVOA compliant services and defines common interfaces for its services. The registry uses common support interfaces to help populate resources in the registry.

- **Capability** - All services define capability metadata, which comprises of XML formatted metadata that describes a particular capability and location of this particular service. The capability also describes what standards this service conforms to. Certain capabilities will be to other VOSI interfaces or the VAMDC-TAP service location along with what standard interface it supports for VAMDC-TAP. Registry uses this VOSI location of the capability metadata to property fill out the resource in the registry. If other VOSI locations are present such as Table and Application metadata it additionally harvests that data.

- **Table Metadata** - Another VOSI interface in XML form to describe table metadata for Catalogue services. Not used within VAMDC

- **Application Metadata** - Not part of VOSI, an extension created to have a piece of XML VOSI for application description. Not used within VAMDC

- **Availability** - Not used by the registry, but is provided as a Support interface to make retrieve information of uptime and other availability information concerning the service. Is used for service monitoring purposes.

WEB ADMINISTRATION

The End User does not have the capability to access registry via this Web interface. Only Scientist and other Technical users of VAMDC can use the registry to add or update resources in the VAMDC registry. End Users use other client programs such as the Astrogrid VODesktop to query on the resources located inside the registry.

2.1 Location, home page and menus

Production registry: http://registry.vamdc.eu/registry-12.07/

Registry of legacy system: http://registry.vamdc.org/registry-11.12/

Development registry: http://casx019-zone1.ast.cam.ac.uk/registry/

2.1.1 Welcome page

The First page is simply a 'Welcome' page to provide access to the capabilities of the registry in the menu items. The Welcome page displays the AuthorityID setup by this registry.

Within the scope of this User Guide only the Querying/Creating/Updating resources are considered (see *Creating a resource*). The other administration parts of the registry are discussed in the Administrator section of the registry.
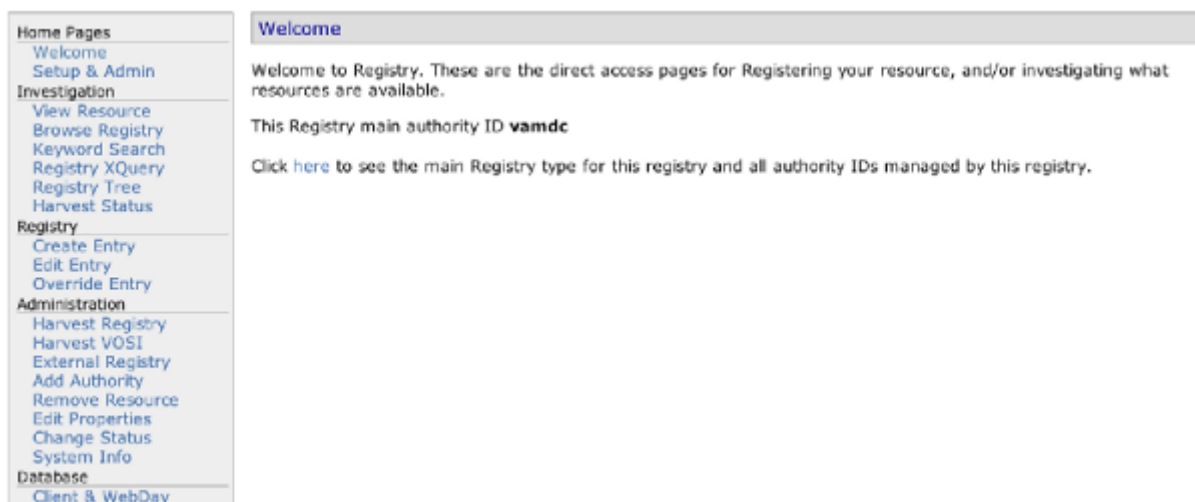

Figure 2.1: Welcome

2.1.2 Creating a resource

Choosing 'Create Entry' will begin creating a new resource, you must choose a unique identifier for this registry and the 'type' of resource. (See *Creating a resource*)
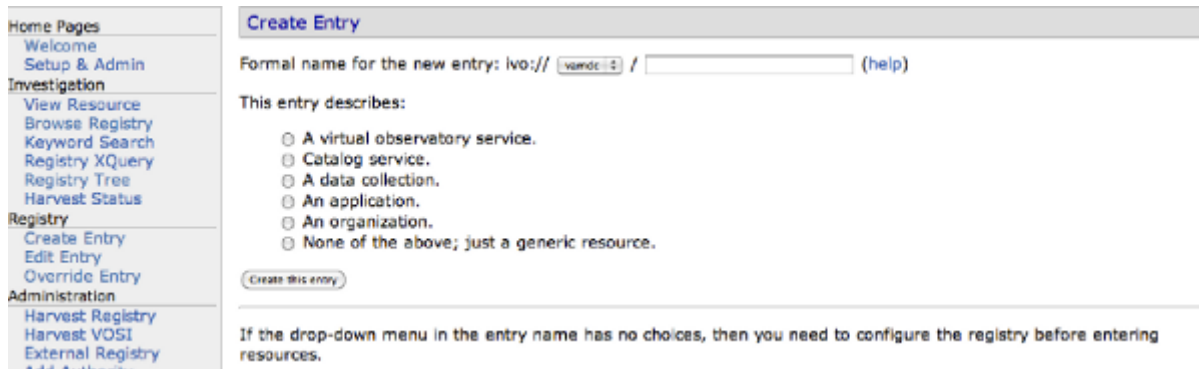

Figure 2.2: Creating a resource

2.1.3 Core Data

The Core Data as shown in Figure 3 must be filled out prior to creating a resource. It is the Core part of the resource XML, which is entered into the registry. A 'help' link is provided next to each field to help enter the data. It is desirable to open the help link as a new tab or window.

2.1.4 Browsing the registry

Several menu options exist to query or investigate the registry, however the most commonly used option is 'Browse'. When initially clicked, all the resources in the registry are displayed. It is possible to filter by identifier if needed. Only selected information is shown about the resource on the Browse screen including title, type of resource, and identifier (see *Browse*). It may be necessary to select to perform a particular action or task on the resource such as updating, viewing, or see the raw XML.

2.1.5 Update resource option

On the Browse registry page 'Edit' link in the Query pages (shown in *Browse*) is available to fulfil the options of resource detail update. A choice of Updating the Core Information, Updating other information via the VOSI interface, Updating Coverage information, and finally an Update Screen that gives access to the Raw XML are available.

2.1.6 Edit core

Similar page as 'Creating a resource' with the html form fields populated with what is contained in the registry is shown in *Updating Core metadata*.

IVO identifier	ivo://vamdc_test/vamdc_testentry	
Resource status	active	help
Title		help
Publisher's name		help
Publisher's IVO identifier		help
Creator's name		help
Creator's IVO identifier		help
URL of creator's logo		help
Release-date of resource		help
Version of resource		help
Name of contact person		help
Postal address of contact person		help
Email address of contact person		help
Telephone number of contact person		help
Keywords describing this resource		help
Text describing this resource		help
Source of the resource content		help
URL for web page describing this resource	http://www.astrogrid.org/	help
Type of the resource content	Other	help
Intended audience	Research	help
WebBrowser Capability URL		help

Record this information in the registry

Figure 2.3: Core Data

Figure 2.4: Browse

Figure 2.5: Edit Choice

Figure 2.6: Updating Core metadata

2.1.7 Using VOSI

After Creating or during Updating a service an option is be given to update the service with a VOSI capabilities location. This reads the XML metadata and populates the resource accordingly (shown in *Populating resource based on VOSI/Capabilities*). This Form expects a VOSI URL to point to 'Capabilities' URL that describes locations and standards of this service and VOSI services (see *Populating resource based on VOSI/Capabilities*).

Figure 2.7: Populating resource based on VOSI/Capabilities

2.1.8 Editing an existing resource (raw XML)

Populating resource based on VOSI/Capabilities demonstrates the ability to upload Raw XML, local file, URL location or using an html Text box. Options shown in *Using raw XML to update or create resources* may be useful if an XML resource is already locally saved, one may then edit manually and directly upload a new update. This option is also useful for making quick changes. When submitted it is validated and placed into the registry.

Note: If you change the identifier to something that is not in the registry it will automatically create the entry in the registry.

Figure 2.8: Using raw XML to update or create resources

2.2 Video tutorial

You can download a video tutorial for using the registry administration web pages here .

QUERYING

Querying requires software to conform to the IVOA registry interface specification: http://www.ivoa.net/Documents/RegistryInterface/ . There are four main query methods:

- Xquery - not supported in all IVOA registries, but VAMDC does support Xquery. It is the most advanced way of querying on the registry. But requires knowledge of all the registration schema structure to construct certain XPATH nodes. Software such as Astrogrid VODesktop gives you a simple query interface and performs the more complex Xquery behind the scenes.

- ADQL - also an advanced way of querying the registry. It is an SQL form over XML. It does not have all the functionality as Xquery such as ADQL can only return the full resource from the registry, but is standard and all IVOA registries conform to ADQL. Software that wants to be certain to work for all IVOA registries tend to use this method such as TopCat.

- KeywordSearch - generic keyword search mechanism.

- GetResource - gets one particular resource entry from the registry, knowing that resource IVOA identifier.

See *Querying the Registry for VAMDC Resources* for the available libraries and help querying of the registry.

3.1 XQuery

The recommended way to look for things in the registry is to send in queries in the XQuery language. The registry responds with XML documents carrying the information matching the query.

For a given XQuery and for a given programming language, the details of the query can be encapsulated in a client library; the library phrases the query based on simple parameters to a method call. This has been done for typical VAMDC queries from Java, and the library is described below. Often this is all you need, but sometimes it is easier or more efficient to make the query directly from your application code.

If you do not understand the basics of XQuery you will not understand the details of this section. Either skip ahead to the descriptions of the client library or have a look at an XQuery tutorial.

This is an example of a registry XQuery. It finds the formal names of all the VAMDC-TAP services.

```
declare namespace ri='http://www.ivoa.net/xml/RegistryInterface/v1.0';
for $x in //ri:Resource
where $x/capability[@standardID='ivo://vamdc/std/VAMDC-TAP']
and $x/@status='active'
return $x/identifier
```

The query could be translated as "Find all the registration documents containing a capability with the VAMDC-TAP identifier, taking only those for active services; give me the IVORNs and throw the rest away". The XPath construct `//ri:Resource` means "all the registration documents". Because this searches for a type of element, and because types have namespaces, we have to map the namespace to a prefix (the first line) and use that prefix in specifying the type (the `ri:` in `ri:Resource`).

The registry's response will be be a document containing identifier elements as immediate children of the document element.

Most queries will be in this general form. It is important to restrict the search to active resources because the registry contains some that are "inactive" (resting, pending refurbishment) or "deleted" (gone for good, but not actually removed from the registry database).

This is a possible rearrangement of the query above.

```
declare namespace ri='http://www.ivoa.net/xml/RegistryInterface/v1.0';
return //ri:Resource[capability[@standardID='ivo://ivoa.net/std/TAP' and @status='active']/identi
```

The constraints have been moved inside the square brackets in the return clause and the where clause disappears. Both queries should raise the same results; you can use whichever form is easiest for you.

Here is a different query, searching for TAP services.

```
declare namespace ri='http://www.ivoa.net/xml/RegistryInterface/v1.0';
for $x in //ri:Resource
where $x/capability[@standardID='ivo://ivoa.net/std/TAP']
and $x/@status='active'
return $x
```

Here, the identifier for the capability is different - IVOA TAP instead of VAMDC-TAP - and, more importantly, the query returns all the parts of the registration documents, not just the identifiers.

As a final example, here is a query to give the access URLs (the URLs to which you would send the data query) for VAMDC-TAP services that can return data on measured wavelengths of radiative transitions.

```
declare namespace ri='http://www.ivoa.net/xml/RegistryInterface/v1.0';
for $x in //ri:Resource
where $x/capability[@standardID='ivo://vamdc/std/VAMDC-TAP'
and returnable='RadTransWavelengthExperimentalValue']
and $x/@status='active'
return $x/capability[@standardID='ivo://vamdc/std/VAMDC-TAP']/interface/accessURL
```

The new trick here is to have a constraint - the part in square brackets - in both the where clause and the return clause. The constraint in the WHERE clause finds the right registrations and the one in the return clause makes sure that we get the URLs only from the VAMDC-TAP interfaces and not from any other interfaces those services might have.

The search term `RadTransWavelengthExperimentalValue` comes from the VAMDC dictionary. It appears in the query because VAMDC-TAP service register their returnables using that dictionary. The term is not inherent to XQuery or to the registry.

QUERYING THE REGISTRY FOR VAMDC RESOURCES

There are, broadly, four ways to put a query to the registry from Java. In increasing order of abstraction and preference they are:

1. Call the registry (SOAP) web-service directly.

2. Use the AstroGrid client library.

3. Use the AstroGrid Astro-Runtime API.

4. Use the VAMDC client library.

The AstroGrid client library is worth considering. If you have a simple query (e.g. if you already know the identifier for the service of choice and just want to extract the access URL) then the library is quite good. If you have a more-general query, particularly one that will return results from more than one registration, then the library has to be forced into a non-standard configuration to work properly.

The Astro Runtime is a better abstraction for the registry and is actually intended for applications programmers (the AstroGrid client-library above is aimed at system engineers). It can return results as Java objects rather than as XML, which is sometimes easier to deal with. However, you have to write your own query text, typically in XQuery. There is a VAMDC client-library (see below), which tries to abstract common queries so you don't need to write any XQuery text. This library knows about (some of) the service types important in VAMDC. Support for forming queries is good. Support for parsing the results is limited; you either get a DOM or simple values in strings, depending on the kind of query.

4.1 VAMDC registry query library

A small (single-class) library is available for VAMDC work. Version 3.0 of this library as well as a zip file containing all the third-party, supporting jars are available for download from the links below. (The AstroGrid client-library for the registry is one of the third-party jars if you want to use it directly.)

- VAMDC registry-client library version 3.0

- 3rd-party jars supporting the registry-client library

Some usage notes follow. For the full range of function, see the Javadoc. Other technical descriptions of the software are available, but the main documentation is this page and the Javadoc.

To use the library, instantiate the single class `eu.vamdc.registry.Registry`. Each method call makes one registry query (internally, some of them make a sequence of queries but you receive a single set of results). You can reuse the object for multiple, successive queries, but it is not safe to share it between threads. The no-argument constructor makes a client for the release registry. To use the development registry, pass the constant `Registry.DEVELOPMENT_REGISTRY_ENDPOINT` to the constructor (that is a string literal stating the endpoint for the registry of choice). The ability to select the development registry was added in v2.0 of the client.

The library lets you query for three kinds of information: whole registration documents, IVORNs and access URLs. The latter two types are delivered as lists or sets of strings and the registration documents as

org.w3c.dom.Document instances. In the documents, the document element is an uninteresting wrapper and the query results are its first-level children.

Here is an example of finding all the TAP services (this matches one of the XQuery examples in the section above):

```
import eu.vamdc.registry.Registry;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
...
    Registry reggie = new Registry();
    Document results = reggie.findTap();
    NodeList nl = results.getDocumentElement().getElementsByTagName("ri:Resource");
    for (int i = 0; i < nl.getLength(); i++) {
        // Do something with this registration document...
    }
```

You could also dismantle the results document using XSLT or XPATH. This might be better than using the DOM API.

If you want all the information for all the VAMDC-TAP services in the registry, there is a convenience method:

```
import eu.vamdc.registry.Registry;
Registry reggie = new Registry();
Document results = reggie.findVamdcTap();
```

Sometimes you just want the access URLs for a class of services. Here is how:

```
import eu.vamdc.registry.Registry;
import java.net.URL;
import java.util.Set;
...
    Registry reggie = new Registry();
    Set<String> results = reggie.findAccessUrlsByCapability(Registry.VAMDC_TAP_ID);
    for (String s : results) {
        URL u = new URL(s);
        // Use this service...
    }
```

Note the use of a string constant to set the standard-identifier for VAMDC-TAP. You could also write the literal identifier: `ivo://vamdc/std/VAMDC-TAP`.

If you want to select resources by special criteria, then you have to supply your own XQuery. Using the last example from the XQuery section above, this code looks for the access URLs of VAMDC-TAP services that can give wavelength data.

```
import eu.vamdc.registry.Registry;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
...
    Registry reggie = new Registry();
    String query =
        "declare namespace ri='http://www.ivoa.net/xml/RegistryInterface/v1.0'; " +
        "for $x in //ri:Resource " +
        "where $x/capability[@standardID='ivo://vamdc/std/VAMDC-TAP' " +
        "and restrictable='AtomSymbol'] " +
        "and $x/@status='active' " +
        "return $x/capability[@standardID='ivo://vamdc/std/VAMDC-TAP']/interface/accessURL";
    Document results = reggie.executeXquery(query);
    //    NodeList nl = results.getDocumentElement().getElementsByTagName("ri:Resource");
    NodeList nl = results.getDocumentElement().getElementsByTagName("accessURL");
    for (int i = 0; i < nl.getLength(); i++) {
        // Do something with this information...
        System.out.println(nl.item(i).getFirstChild().getNodeValue());
    }
```

Note the spaces at the end of each fragment of the query: these are necessary to make the overall query correct.

4.1.1 Sample registry query project

Some sample query routines are demonstrated in this eclipse project: registry-query-sample-project.tar.gz

Routines are:

- Collection getIVOAIDs() - get all IVOA identifiers for TAP-VAMDC services
- String getTapURL(String ivoaid) - get access URL for specific service
- Collection getRestrictables(String ivoaID) - get list of supported restrictables for specific service

4.2 Lightweight VAMDC Registry Client

Another implementation of VAMDC registry client library exists, used by VAMDC Portal and TAPValidator tool.

This client has some specific features:

- Higher-level client, instantly provides VAMDC-TAP and XSAMS Processor service access URLs.
- Uses JAXB mappings for passing registration information.
- Does one registry lookup at instantiation time, then uses cached results.
- Does not require any legacy Astrogrid library code, smaller dependencies footprint.

It will be available for download from VAMDC official libraries section later on.

4.3 VAMDC web-portal

If you are only interested in getting data from VAMDC-TAP data-services, then you can access them through VAMDC web-portal, which handles registry lookups by itself.

4.4 Astrogrid VODesktop

This user guide only shows how to point to the VAMDC registry with Astrogrid VODesktop and the main query screen for the registry.

http://www.astrogrid.org

4.4.1 StartUp

When VODesktop is launched, the first screen is normally VOExplorer. You can also find VOExplorer by selecting Window -> New VOExplorer in the menu. VOExplorer allows you to search the registry for resources in the registry. Once you select a resource you can View its contents and perform certain actions that VODesktop might be aware of such as querying a Catalogue Service or running a particular Application.

Clicking the 'New Smart List' button brings up a window to begin searching on the registry. As the Text Boxes are filled out it queries registries for a 'count' of how many resources would be returned, and allow making the decision to perform the query or add new constraints.
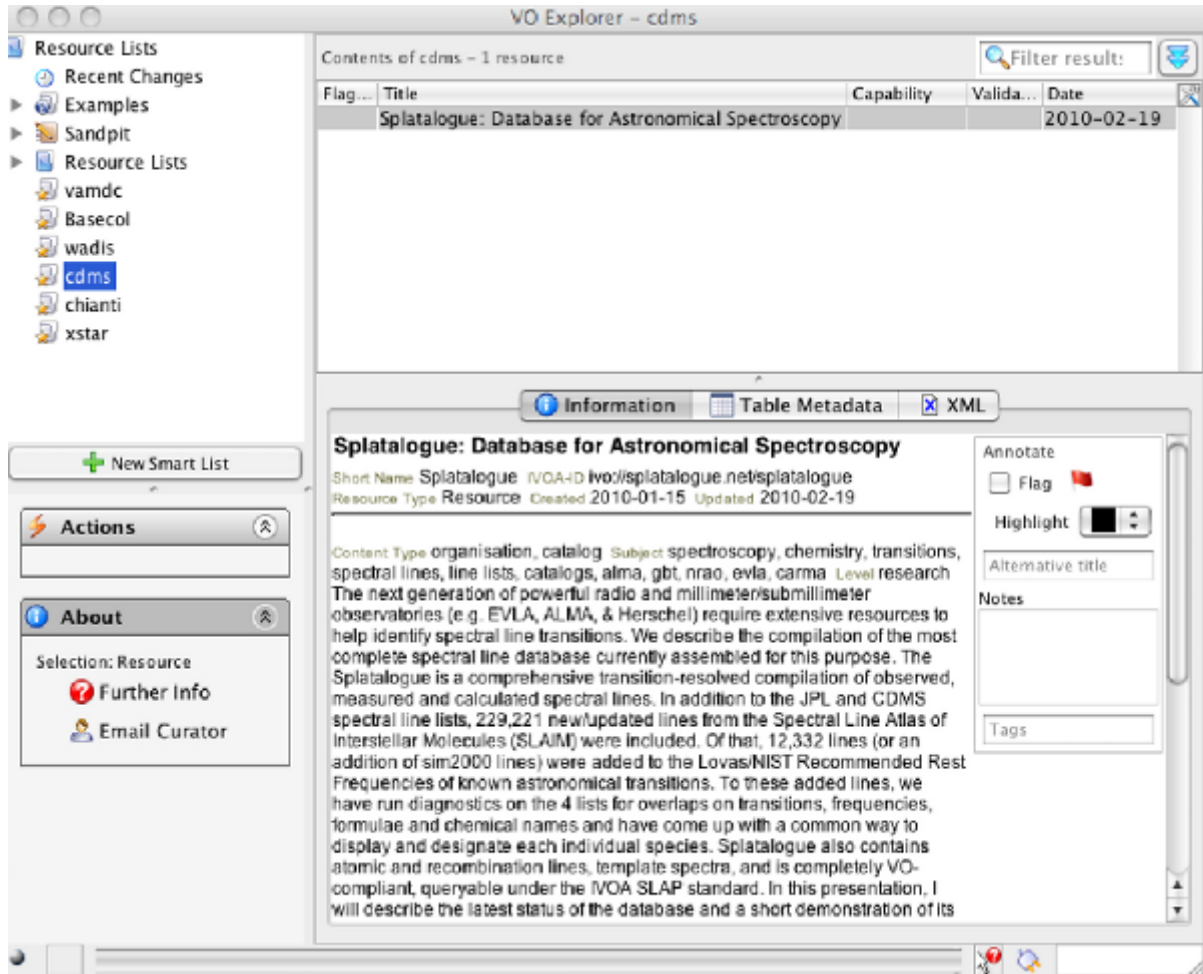
Figure 4.1: Search registry window
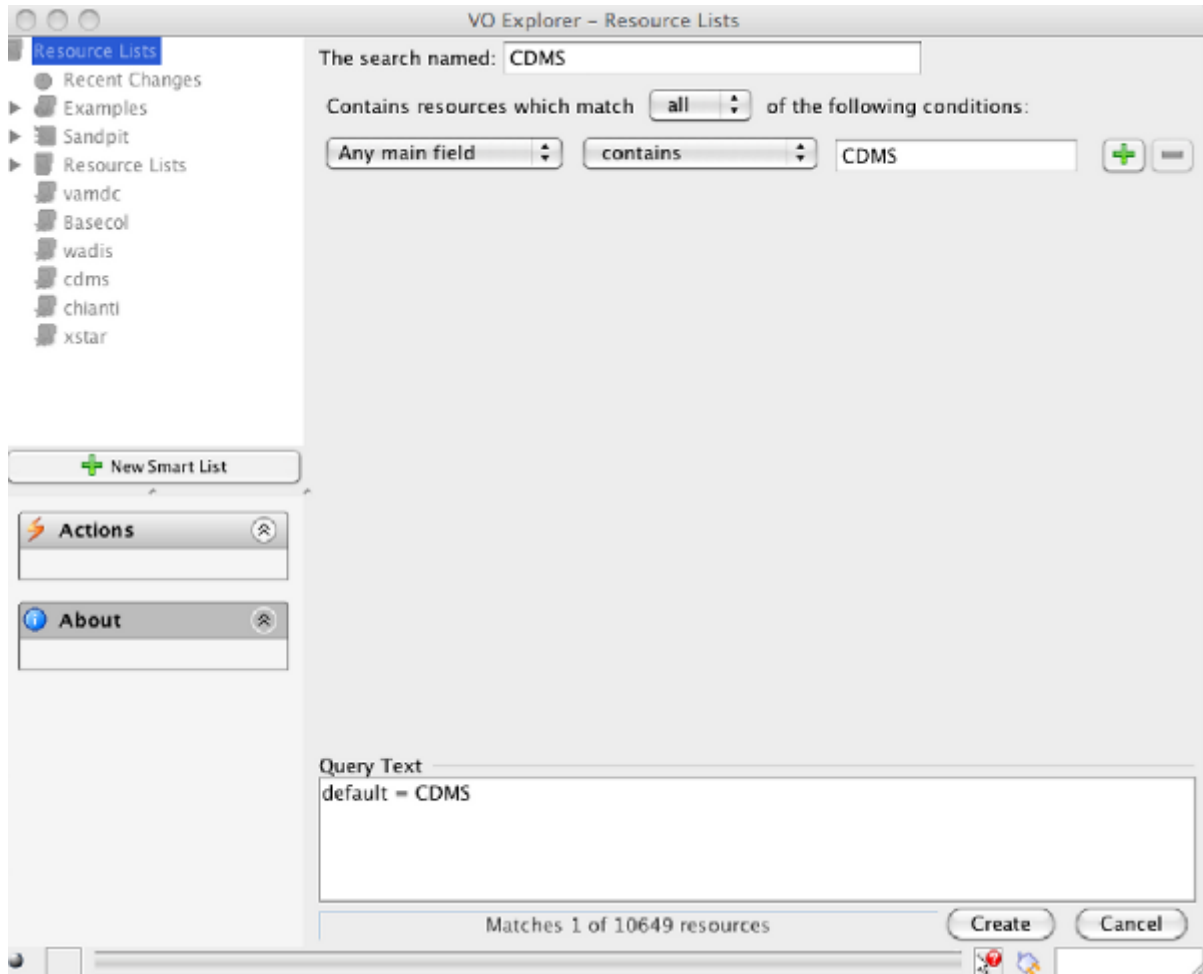
Figure 4.2: Resource list window

4.4.2 Preferences

In the case of not being able to find any VAMDC resources it is possible that you located an incorrect registry. By clicking on VODesktop->Preferences brings up a window that allows switching to a different registry. Ensure that the correct VAMDC registry is selected (pointed to).

Production registry: http://registry.vamdc.eu/vamdc_registry/services/RegistryQueryv1_0

Development registry: http://casx019-zone1.ast.cam.ac.uk/registry/services/RegistryQueryv1_0
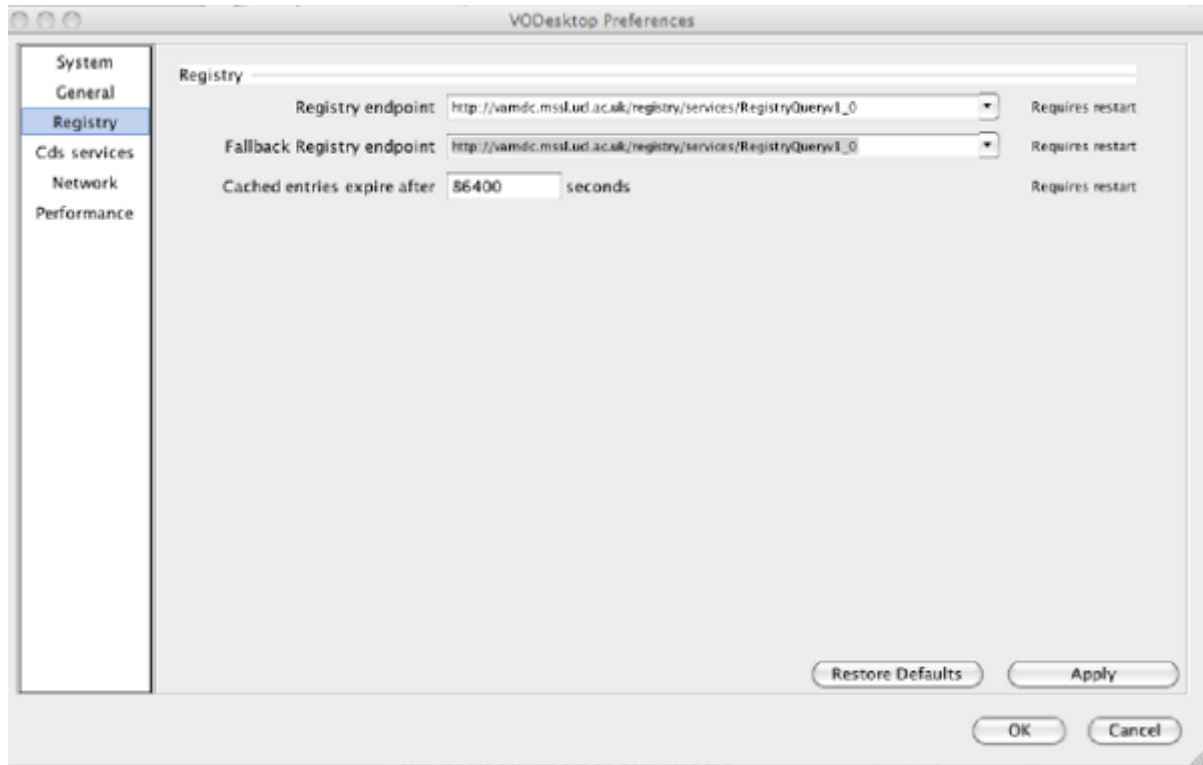
Figure 4.3: VO Preferences

REGISTERING VAMDC-TAP

5.1 Registration process

If you are a new user you must first get a password to register your service in the registry. For passwords contact registry @ vamdc.eu .

The registration process is as follows:

1. Install VAMDC-TAP service on a web visible server.

2. Go to registry web user interface and register your service core data. (As explained in *Web Administration*)

3. Ask the registry to load VOSI data from your service. You invoke this from the registry's web user-interface (see *Web Administration*) and the registry reads the capability data described below. You have to provide the URL for the VOSI data. In a typical VAMDC-TAP service this will be a URL ending in `tap/capabilities`; e.g.:

   ```
   http://vald.astro.uu.se/tap/capabilities
   ```

5.2 Structure of registration document (capabilities)

The registry contains an XML registration-document for each known service (and for a few things that are not strictly services, but we ignore them here).

Inside each service registration-document there are capability elements defining the kinds of service provided. Inside each capability is an interface (sometimes more than one) and inside each interface is an accessURL element the value of which defines where to find that part of the service.

A typical registry query from code looks only at the capabilities and ignores the rest of the registration.

Here's an example of a capability:

```
<capability standardID="ivo://ivoa.net/std/TAP">
    <interface xsi:type="vs:ParamHTTP">
        <accessURL use="base">http://vald.astro.uu.se/tap/</accessURL>
    </interface>
</capability>
```

This refers to a web service on the Uppsala mirror of VALD.

Note the standardID attribute: the value of this identifies this capability as IVOA's Table Access Protocol (TAP) web-service. Capabilities for standard protocols always have a standardID attribute to tell them apart.

The interface element has `xsi:type="vs:ParamHTTP"`, meaning that basic HTTP GET or POST work on this interface, and that HTTP parameters are involved in the protocol. Other types are `vs:WebService`, meaning a SOAP endpoint, and `vr:WebBrowser`, meaning a web site for interactive viewing. The accessURL element identifies a web-resource on a server in Uppsala. The use="base" attribute means that the client must add a suffix to the given URL to get a working URL for a query. The nature of the suffix is defined by the protocol

identified in the standardID attribute of the capability. Here, because it is TAP, we know to add `/sync?` and then the HTTP parameters defining a query. Here is another example, from the same registration.

```
<capability standardID="ivo://vamdc/std/VAMDC-TAP"
    xmlns:tx="http://www.vamdc.org/xml/VAMDC-TAP/v1.0" xsi:type="tx:VamdcTap">
    <interface xsi:type="vs:ParamHTTP">
        <accessURL use="base">http://vald.astro.uu.se/tap/</accessURL>
    </interface>
    <returnable>AtomStateLandeFactorRef</returnable>
    <returnable>AtomNuclearCharge</returnable>
    <returnable>SourceCategory</returnable>
    ...
    <restrictable>AtomStateEnergy</restrictable>
    <restrictable>AtomNuclearCharge</restrictable>
    <restrictable>RadTransLogGF</restrictable>
    <restrictable>AtomSymbol</restrictable>
    <restrictable>RadTransWavelengthExperimentalValue</restrictable>
    <restrictable>AtomIonCharge</restrictable>
</capability>
```

This is the VAMDC-TAP capability for the same VALD mirror. It is very similar to the TAP example (VAMDC-TAP being a specialization of TAP); in fact the access URL is the same.

The main difference is the returnable and restrictable elements following the interface element. The returnables tell you which quantities can be obtained from this service. The restrictables tell you which terms can be used as contraints in the query (i.e. which column names can appear in the WHERE clause of a query).

The standardID value identifies this capability as VAMDC-TAP. The `xsi:type` attribute identifies the structural type as one for which the XML schema allows the returnable and restrictable children.

In both these examples, and in all capabilities you are likely to see, the elements are in the default namespace. This means that they are written without a namespace prefix, and you do not state a namespace when searching for elements by their names. However, some of the types have specific namespaces; if you search for elements by type you will have to deal with their those.

Please note the exact namespace used for the VAMDC-TAP example, above: `http://www.vamdc.org/xml/VAMDC-TAP/v1.0`. Earlier examples used the namespace `http://www.vamdc.eu/xml/TAPXSAMS/v1.0` which is no longer valid.

REFERENCES

- Top level documents and schemas: http://www.ivoa.net/Documents/

- IVOA identifiers: http://www.ivoa.net/Documents/latest/IDs.html

- Registry interface: http://www.ivoa.net/Documents/RegistryInterface/

- VOResource core schema: http://www.ivoa.net/xml/VOResource/VOResourcev1.0.xsd

- VODataService schema describe catalogue services: http://www.ivoa.net/xml/VODataService/VODataService-v1.1.xsd

- VORegistry schema describe registry services: http://www.ivoa.net/xml/VORegistry/VORegistryv1.0.xsd

- ADQL one of the query languages of the registry interface: http://www.ivoa.net/xml/ADQL/ADQL-v1.0.xsd

- Production registry: http://registry.vamdc.org/registry-12.07/

- Legacy production registry: http://registry.vamdc.org/registry-11.12/

- Development registry: http://casx019-zone1.ast.cam.ac.uk/registry/

CHANGE LOG FOR REGISTRY GUIDE

7.1 Changes between v11.05 and v11.12

The `standardID` value identifying the capability for VAMDC nodes has changed from `ivo://vamdc/std/TAP-XSAMS` to `ivo://vamdc/std/VAMDC-TAP`. The examples of querying have been updated accordingly.

The namespace of the schema ruling the VAMDC-TAP capability has changed from `http://www.vamdc.eu/xml/TAPXSAMS/v1.0` to `http://www.vamdc.org/xml/VAMDC-TAP/v1.0`. The section on registering services have been updated accordingly.

The recommended version of the VAMDC client-library for the registry is changed from 2.0 to 3.0.

The sample project associated with the guide used some out-of-date constants. These have been updated to the current standard.